EESTI MAAÜLIKOOL
Tehnikainstituut

Raido Kaju

ELEKTRIENERGIA BÖRSIHINDA ARVESTAV ELEKTRIKÜTTE JUHTSEADE

ELECTRICAL ENERGY PRICE BASED HEATING CONTROL UNIT

Bakalaureusetöö
Tehnika ja tehnoloogia õppekava

Juhendaja: nooremteadur Heiki Lill, MSc

Tartu 2021

Eesti Maaülikool
Estonian University of Life Sciences

www.emu.ee
Eesti Maaülikool
Estonian University of Life Sciences

Eesti Maaülikool Kreutzwaldi 1, Tartu 51006		Bakalaureusetöö lühikokkuvõte	
Autor: Raido Kaju		Õppekava: Tehnika ja tehnoloogia õppekava	
Pealkiri: Elektrienergia börsihinda arvestav elektrikütte juhtseade			
Lehekülgi: 30	Jooniseid: 14	Tabeleid: 9	Lisasid: 3
Osakond / Õppetool: Energiakasutuse õppetool ETIS-e teadusvaldkond ja CERC S-i kood: 4. Loodusteadused ja tehnika 4.17. Energeetikaalased uuringud T140 Energeetika Juhendaja(d): nooremteadur Heiki Lill, MSc Kaitsmiskoht ja -aasta: Tartu 2021			

Inimeste elukeskkond on elektrienergiast suurel määral sõltuv ja mõjutatav. Tänapäeval saab öelda, et ilma elektrienergiata on väga keeruline hakkama saada, kuna enamus tegevusi toimivad selle abil või sellega kooskõlas. Bakalaureusetöö eesmärk on koostada toimiv börsihinda jälgiv ja sellest juhinduv seade kütteseadme töö optimeerimiseks. Juhtseade on seadistatud toimima nii, et kütteseade toimiks toimiks vaid keskmisest börsihinnast soodsamatel aegadel. Töö käigus sai testitud seadme toimivust ja vaadeldud energianandmeid mis selle käigus sai kogutud. Kogutud andmed viitasid seadme korrektsele toimivusele. Elektrikulu sama börsiperioodiga võrreldes oli juhtimata seadme puhul 6.17% kõrgem kui börsihinnaga juhitud seadme puhul. Seadme juhtimiskoodi on võimalik muuta nii, et seade kasutaks veel optimaalsemalt ära hinnainfot ja rakendaks seadet effektiivsemalt ka keskmisest soodsamatel aegadel.

Märksõnad: elektrienergia, börsihind, kulude optimeerimine

www.emu.ee
Eesti Maaülikool
Estonian University of Life Sciences

Estonian University of Life Sciences Kreutzwaldi 1, Tartu 51006		Bachelor's Thesis	
Author: Raido Kaju		Curriculum: Engineering and technology curriculum	
Title: Electrical Energy Price Based Heating Control Unit			
Pages: 30	Figures: 14	Tables: 9	Appendixes: 3
Department / Chair: Energy Engineering Field of research and (CERC S) code: 4. Natural Sciences and Engineering 4.17. Energetic research T140 Energy research Supervisors: Juunior Research Fellow Heiki Lill, MSc Place and date: Tartu 2021			

People´s living environment is highly dependent on and affected by electricity. Today, it can be said that it is very difficult to manage without electricity, because most activities work by using it. The aim of the bachelor's thesis is to compile a functioning device that monitors and is guided by the stock exchange price in order to optimize the operation of the heating device. The control unit is set to operate in such a way that the heating unit only operates at times better than the average stock market price. During the work, the performance of the device was tested and the energy data collected during it were observed. The collected data indicated the correct performance of the device. Compared to the same exchange period, the electricity consumption for a non-controlled device was 6.17% higher than for an exchange-driven device. The control code of the device can be changed so that the device makes even more optimal use of price information and implements the device more efficiently even in more favorable times than average.

Keywords: electricity, stock price, cost optimization

SISUKORD

TÄHISED JA LÜHENDID 5
SISSEJUHATUS 6
1. ELEKTRITURG 8
2. ELEKTROONIKA 10
2.1. Kasutatud komponendid 10
2.2. Optimeerimisseadme majanduslik kulu 15
3. SEADME JUHTIMINE 16
4. JUHTIMISE TEOSTAMINE 18
4.1. Nord Pool Spotist informatsiooni hankimine ja töötlemine 18
4.2. Juhtseadme otsuse tegemine 19
4.3. Otsuse põhjal relee käivitamine/sulgemine 20
5. TULEMUSTE ANALÜÜS 23
5.1. Energiakulu seadmeta 23
5.2. Energiakulu seadmega 24
5.3. Majandusanalüüs 25
KOKKUVÕTE 29
KASUTATUD KIRJANDUS 31
LIHTLITSENTS 33
LISAD 34

TÄHISED JA LÜHENDID

Alalisvool – vool, mille suund ei muutu;
GPIO – Raspberry Pi lühend üldotstarbelistest sisenditest/väljunditest;
Nord Pool Spot – ettevõte, mis peab Põhjmaade ühist elektribörsi;
Relee – seade, mis impulsi abil ühendab vooluvõrgu;
Vahelduvvool – elektrivool, mille suund perioodiliselt muutub.

SISSEJUHATUS

Alates elektri avastamisest on inimkond püüdnud selle energiat rakendada. Inimeste elukeskkond on elektrienergiast suurel määral sõltuv ja mõjutatav. Selle olemasolu on läbi aastate tõusnud inimeste elus üha olulisemale kohale. Tänapäeval saab öelda, et ilma elektrienergiata on väga keeruline hakkama saada, kuna enamus tegevusi toimivad selle abil.

Eesti kuulub aastast 2013 Nord Pool Spot elektri piirkonda. See on avanud eesti elektrituru börsihinnaga kauplemiseks. Nord Pool Spotis määratakse elektri hind igaks tunniks eraldi. See tähendab, et järgneva 24 h börsihind avaldatakse tunni lõikes eraldiseisvalt. Vastaval perioodil, kui elektrienergia tarbimine on suurem, on ka börsihind vastavalt kõrgem. Sellest lähtuvalt on peamiselt öösel elektri börsihind madalam. Elanikkond puhkab ja suuremat elektri tarbimist ei toimu. Sellest hoolimata on hinnagraafik ja börsihind muutuvad. Käesolevaid erinevusi on võimalik ära kasutada kütteseadmete hinna efektiivsemaks juhtimiseks, kasutades kütteseadet tundidel, mil elektrienergia hind on madalam.

Energia tootmisega ja tarbijani transportimisega kaasnevad alati nii lühi- kui ka pikemaajalised kulutused. Selle põhjal kujuneb lõpphind. Energiatarbimise kasv kajastub ka energiakulude kasvus. Börsihinnal baseeruvad elektripaketid annavad võimaluse elektrienergia tarbimist mõjutada sellisel viisil, et kulu energia tarbimise eest väheneb. Elektrienergia on enamjaolt soodsam öösiti, kui elektri tarbimine on päevasega võrreldes oluliselt väiksema mahuline.

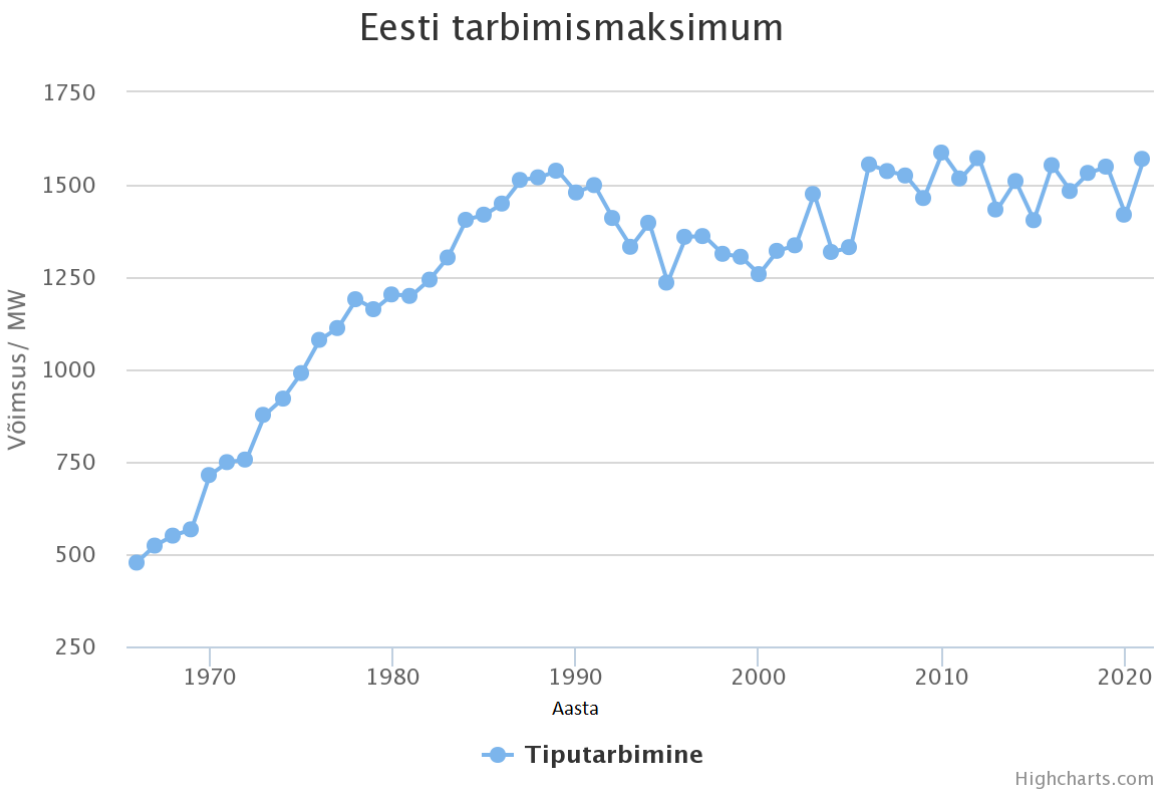
Käesolev lõputöö käsitleb kütteseadme toimivust börsihinna kõikumisi arvesse võttes. Töö eesmärgiks oli koostada elektrienergia börsihinda arvestav elektrikütte juhtseade. Juhtseade sai paigaldatud korterelamu vannituppa, millel on elektripõrandaküte. Köetava ruumi pindala on 7.7 m^2 ja lae kõrgus 3.4 m. Köetava ruumi maht on 25 m^3. Juhtseade sai seadistatud hoidma ruumis keskmisest odavama elektrihinna puhul keskmist temperatuuri 26 °C ja keskmisest kallima hinna puhul keskmist temperatuuri 24 °C. Ilma optimeerimisseadmeta

juhtis küttekeha toimimist Heber HT-125 põrandakütte termostaat. Termostaat sai seadistatud hoidma ruumis temperatuuri 25 °C. Põrandakütte kaabel on tehase andmete järgi 1200 w.

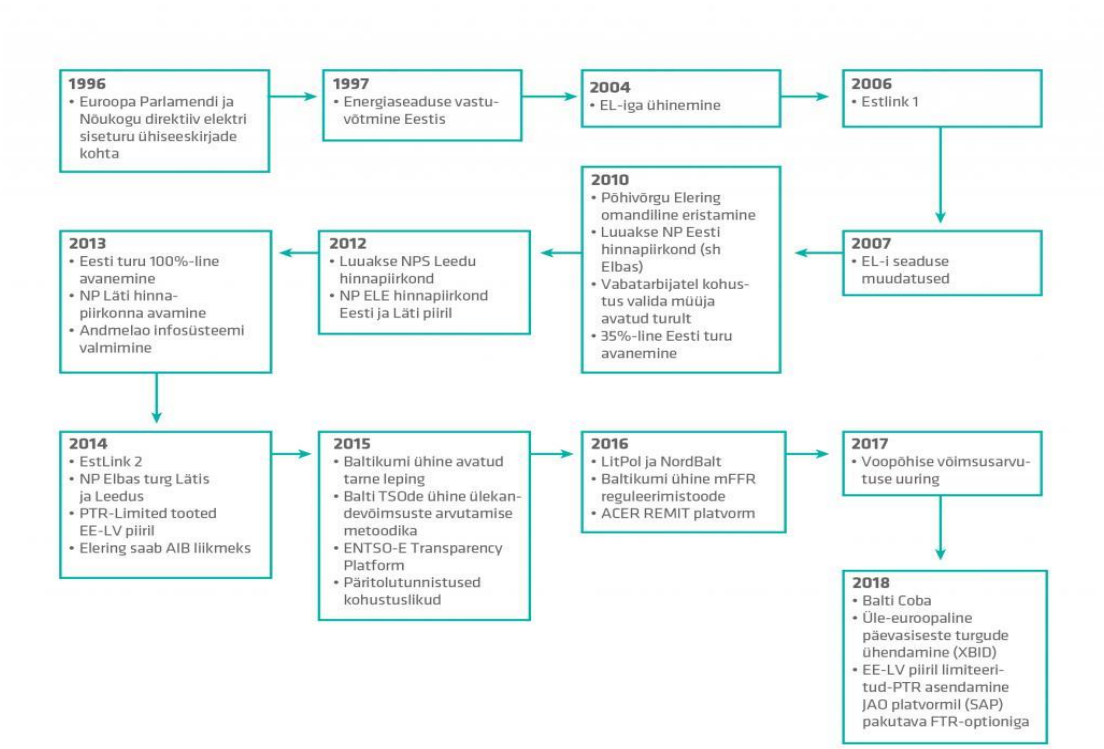
1. ELEKTRITURG

Elektrienergia tarbimine koos võrgukadudega oli 2020. aastal Eestis suurusjärgus 8,44 TWh aastas. Toodang seevastu kõigest 4,8 TWh. See teeb energiabilansiks -3.6 TWh. Selle languse taga on peamiselt CO2 kvootide tõttu põlevkivielektri toodangu langus. Kogu puuduolev elekter tuli sisse osta. Taastuvenergia toodang 2020. aastal oli 2.2 TWh. See on märkimisväärne muutus fossiilsete kütuste pealt taastuvatele energiaallikatele.

Alates elektrienergia kasutuselevõtust on elektri tarbimine aasta-aastalt järjest kasvanud. Elektrienergia tarbimise maksimum on kasvanud jõudsalt aastani 1989. Taasiseseisvumise järel on toimunud väike langus. 2008. aastast edasi on tarbimismaksimum stabiliseerunud. Järgneval joonisel 1 on ära näidatud Eesti elektritarbimine aastate lõikes. Joonis 2 kajastab Eesti elektrituru arengut alates aastast 1996.

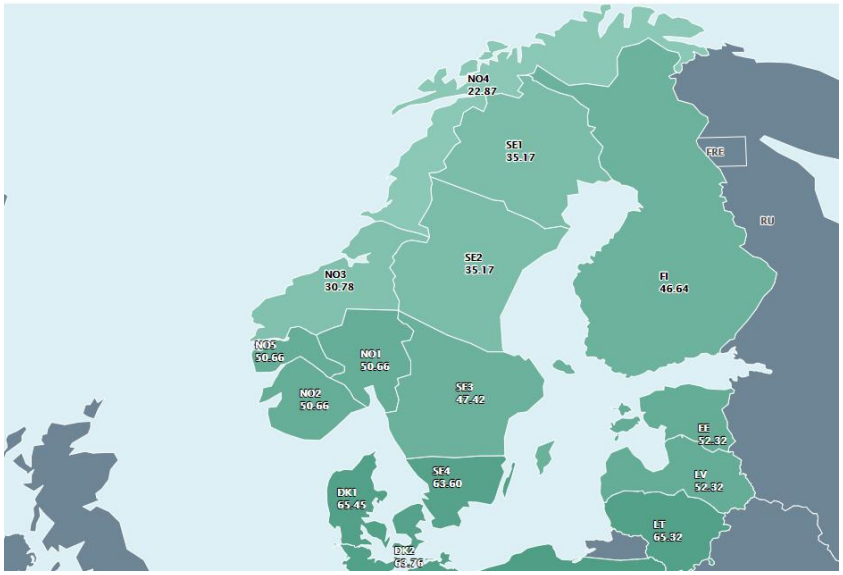


Joonis 1. Eesti tarbimismaksimum



Joonis 2. Eesti elektrituru areng aastatel 1996-2018

Balti riigid moodustavad koos Soome, Rootsi, Taani ja Norraga ühtse põhjamaade elektrituru (Joonis 3). Elektrienergia reguleeritud kaubandust korraldab sellel turul põhjamaade elektribörs Nord Pool. Ühine elektriturg tagab Eesti energia stabiilsuse ja aitab optimeerida elektri tootmist ja tarbimist. Elektriturul on võimalik elektri mahtudega kaubelda ja tarnida puuduolev osa elektrist aegadel kus tarbimine ületab selle hetke tootmist.



Joonis 3. Nord Pool Spot gruppi kuuluvad põhjamaa riigid

2. ELEKTROONIKA

Kütteseadme börsihinnal põhinevaks juhtimiseks on vaja koostada seade, mis on võimeline lugema Nord Pool Spoti andmebaasist ja talletama vastava päeva börsihinda ning käivitama programmi, mis muutuvatest hindadest lähtuvalt juhib kütteseadme tööd. Järgnevalt on välja toodud juhtseadme koostamiseks vajaminevad erinevad komponendid.

2.1. Kasutatud komponendid

Juhtarvuti

Seadme töö juhtimiseks on vajalik juhtarvuti, mis on võimeline võtma kokku erinevate seadmete andmeid ja selle põhjal juhtima relee tööd soovitud tulemuste saavutamiseks.
Käesolevas töös kasutati juhtarvutiks Raspberry Pi 4 Model B seadet. Seadme eelisteks on piisavalt kiire protsessor, küllaldane mälumaht ning sisseehitatud wifi-kaart seadme võrguga ühendamiseks. Samuti on olemas kõik vajalikud liidesed hiire, klaviatuuri ja monitori ühendamiseks. Raspberry Pi peamisteks eelisteks on kompaktsus ja võimaluste rohkus. Seadet on võimalik kasutada suurel hulgal erinevate seadmete juhtimiseks, ühendades selle külge andureid ja sensoreid. Tabelis 1 on näha Raspberry Pi tehnilised andmed.

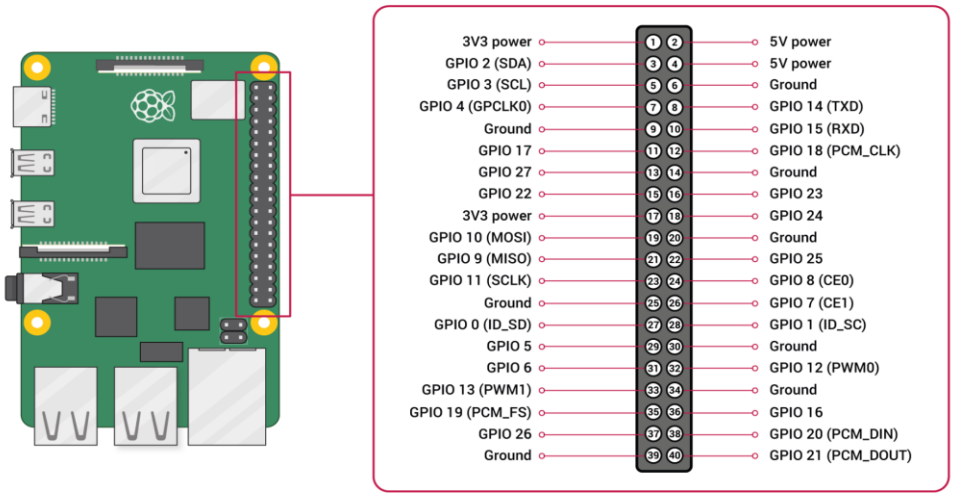
Tabel 1. Raspberry Pi 4 Model B tehnilised andmed

Protsessor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Mälu	2 GB LPDDR4
Ühenduvus	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)

Tabel 1. Järg

Video ja heli	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimeedia	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD-kaardi tugi	Micro SD card slot for loading operating system and data storage
Sisendvool	5 V DC via USB-C connector (minimum 3A1) 5 V DC via GPIO header (minimum 3A1) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Keskkond	Töötemperatuur 0–50º C
Eluiga	The Raspberry Pi 4 Model B will remain in production until at least January 2026.

Joonisel 4 on näha juhtarvuti GPIO viikude tähised.



Joonis 4. Juhtarvuti GPIO viikude tähised

Voolurelee

Voolurelee on digitaalne seadme käivituslüliti, mida on võimalik kontrollida läbi juhtarvuti. Sellega on võimalik juhtida vooluahela tööd. Käesolevas töös kasutati 8 kanaliga 5 V juhitavat releed (Tabel 2). Relee sai valitud põhjusel, et see on Eestis kohapeal saadaval ning sobilik. Raspberry Pi GPIO kaudu juhtimiseks. 8 relee kanalist on kasutusel 1. Ülejäänud

kanalid antud töö käigus kasutust ei leidnud (kuid on võimalik kasutada näiteks boileri kütte juhtimiseks).

Tabel 2. 8-kanaliline 5V releemoodul

Toetatud vooluahel	10 A 250 V DC
Signaali opereerimise pinge	5 V
Seadmete ühendused	Tavaolekus avatud/tavaolekus suletud
Suurus	5.7 x 13.8 cm
Võimalikud kontrollitavad seadmed	8

Temperatuuriandur

Andur annab vajaliku informatsiooni vastava asukoha temperatuurist. See aitab kontrollida seadme ja ruumi temperatuuri. Käesolevasse töösse on valitud laialt levinud digitaalne temperatuuriandur, mille kohta leidub palju seadistamise juhendeid. Selle tehnilised andmed on näha tabelis 3. Eelnimetatud andur on kohalikust kaubandusvõrgust lihtsasti soetatav. Temperatuuriandurite toimimiseks ja info lugemiseks on vaja 5 V ja andmeside ühenduse vahele lisada 4.7-10 kΩ takisti. Takisti abil on võimalik lisada mitu sensorit ühe juhtme alla.

Tabel 3. DS18B20 temperatuurianduri tehnilised andmed

Kasutatav vastavatel temperatuuridel	-55 °C kuni 125 °C
Liidese tüüp	1-wire liides
Sensorit ühe GPIO kanalil	1
+/-0,5° C usaldusvahemik temperatuuridel	-10 °C kuni +85 °C
Reageerimisaeg	Vähem kui 750 ms
Nominaalpinge	3.0 V - 5.5 V

Energiakulu mõõtja

Võimaldab kokku võtta seadme tarbitava energiakulu erinevatel ajahetkedel. Käesolevas töös kasutati energiakulu mõõtjana Shelly 1PM mõõtjat (Tabel 4). Erinevalt konkurendi Sonoff toodetest, on Shelly toodetega võimalik välja tuua energiatarbimise suurusi ühe tunni lõikes

ja eksportide neid redigeeritava failina. Seadme juhtimiseks on loodud äärmiselt lihtne nutitelefoni programm. Selle abil on võimalik jälgida reaalajas kütteseadme elektri tarbimist. Samuti on võimalik saada tarbitud elektri informatsiooni tunni lõikes ja eksportida see redigeeritavas formaadis. See hõlbustab andmete töötlemist ja analüüsimist tarbitud elektri hinnast lähtuvalt. Seade töötab wifi võrgu kaudu ja andmeid on võimalik vaadata üle interneti, ilma, et seadmele oleks ligipääs. Samuti on võimalik seadet samal viisil sisse ja välja lülitada.

Eelnevalt nimetatud aspekt võimaldab seadet kasutada ka lülitina, millega võib ühendatud seadmed üle interneti vooluvõrgust välja lülitada. Selliseid seadmeid on võimalik juhtida läbi Shelly Cloud aplikatsiooni nutitelefonis.

Tabel 4. Shelly 1PM tehnilised andmed

Toiteallikas vahelduvvool	110-230 V ± 10%, 50/60 Hz
Toiteallikas alalisvool	24 – 60 V
Kanalid	1 kanal
Maksimaalne koormus	16 A
Töötemperatuur	-40 °C kuni + 40 °C
Seadme energiatarve	< 1 W
Juhtmevaba/wifi-protokoll	802.11 b/g/n
Raadiosagedus	2400 – 2500 MHz
Raadiosignaali võimsus	1 mW
Raadiosignaali ulatus	kuni 50 m välitingimustes ja kuni 30 m siseruumides (olenevalt ehitusmaterjalidest)
Suurus	41 mm · 36 mm · 17 mm

Vooluadapter

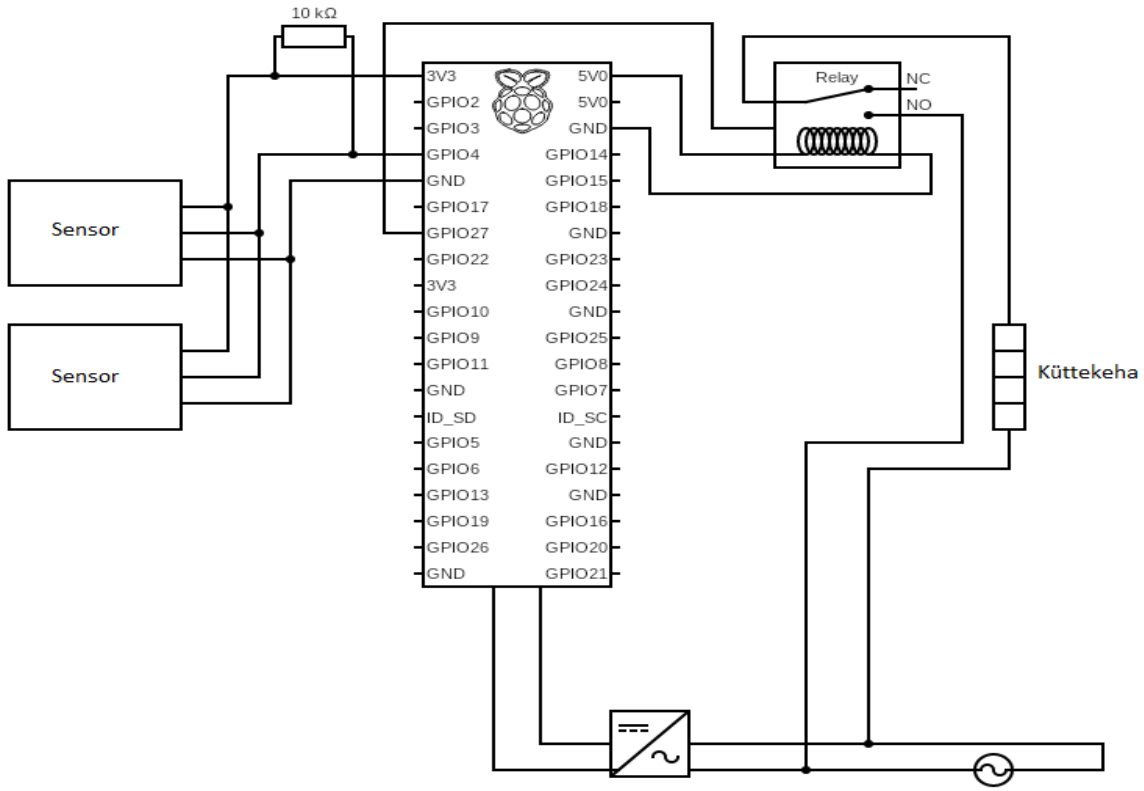
Toitadapter juhtarvuti vooluvõrku ühendamiseks. Tootjapoolne soovitus minimaalselt 3 A väljundvõimsusega USB-C pesaga adapter. Tabelis 5 on kajastatud vooluadapteri tehnilised andmed.

Tabel 5. Stontronics vooluadapteri tehnilised andmed

Mudel	RPI-1BPFCA-05
Sisendpinge	100-240 V
Väljundpinge	5,1 V
Väljundvool	3,0 A

Seadme komponentide ühendamise tarvikud

Erinevad kaablid ja juhtmed vooluvõrgu ühendamiseks ja seadmete omavaheliste ühenduste loomiseks. Micro-HDMI to HDMI kaabel monitori ühendamiseks. Maketeerimisplaat erinevate ühenduste loomiseks. Kogu seadme koosteskeem on näha järgneval joonisel 5.



Joonis 5. Seadme koosteskeem

*Märkus: Sensor = temperatuuriandur; Relay = relee; 10 kΩ suurusega takisti; Raspberry Pi = Juhtseade; NC = Tavaolekus suletud; NO = Tavaolekus avatud

2.2. Optimeerimisseadme majanduslik kulu

Küttekeha börsihinna põhjal reguleerimiseks oli vaja seadet, mis võimaldab hankida internetist informatsiooni ja läbi programmeerimiskeele suudaks käivitada releed, mis juhiks põrandakütte tööd.

Tabel 6. Juhtseadme koostamiseks vajaminevate erinevate komponentide hinnad

SANDISK BY WESTERN DIGITAL MEMORY MICRO SDXC 128GB UHS-IWA	18,68 eur
STONTRONICS - Power Adapter, Raspberry Pi 4 Model B PSU, 5.1 V, 3A,	9,50 eur
Raspberry Pi 4 Model B HDMI Cable, Micro HDMI To HDMI, 1m, Black	8,90 eur
Raspberry Pi 4B, 4x 1,5 GHz,	51,50 eur
10k Ohm takisti	0,20 eur
Maketeerimislaud	2,50 eur
Maketeerimislaua juhtmed	5 eur
Shelly 1PM Elektrikulu mõõtja	20 eur
8-kanaliga relee moodul	14,99 eur
Kokku	**131,27 eur**

Elektrikulu mõõtja juhtimisseadme enda toimimisel vajalik pole. Antud kulutus oleks vajalik vaid optimeerimissüsteemi effektiivsuse mõõtmiseks. Juhtimisseade toimib ka selleta.

3. SEADME JUHTIMINE

Seadme juhtimine on lahendatud programmeerimiskoodi python abil. Seadet on võimalik panna hankima informatsiooni, käivitama erinevaid komponente ja tegema vastavaid arvutusi, mida seadme juhtimiseks vaja.

Esmalt oli vaja koguda andmeid hindade informatsioonist ülevaate saamiseks. Vastavat informatsiooni on võimalik saada Nord Pool Spot leheküljelt. Eelnimetatud leheküljel on palju informatsiooni - ridade ja veergude kaupa näiteks hinnad, kuupäevad. Seal kajastuvad järgmise päeva hinnad. Vastavatest andmetest on võimalik Parse-funktsiooniga õiged andmed välja korjata ja talletada need selleks loodud nordpoolspot_data.json faili, mida juhtprogrammil on lihtsal viisil võimalik kasutada. Parse-funktsiooni abil on võimalik seade panna leidma teatud nimetusi, mis lahtritel on, asendama ebasobivad märgid ja talletama vastava informatsiooni lihtsamini kasutatavas versioonis.

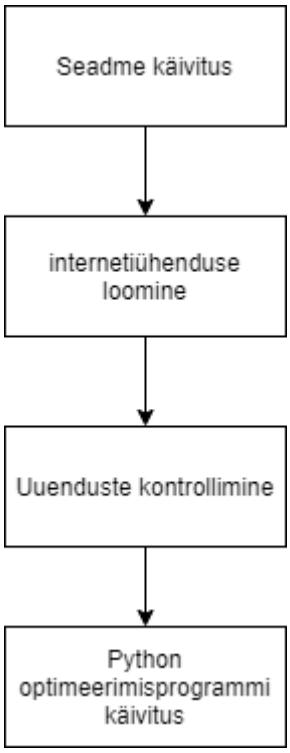
Järgmiseks oli tarvis seadistada temperatuuriandur. Temperatuurianduri seadistamiseks oli vaja avada 1-wire võimalus Raspberry Pi seadistustest. Sensori ühendamise järel ja eelnimetatud võimaluse avamisel, asub temperatuuriandur automaatselt koguma ja salvestama sensorist tulevaid andmeid. Et teada, kuhu ja milliseid andmeid sensor salvestab, tuleb otsida seadmete nimistust vastav sensor oma unikaalse koodiga. Peale temperatuurianduri seadistamist Rapsberry Pi operatsioonisüsteemis on võimalik seadistada DS18B20 temperatuuriandur lugema temperatuuri informatsiooni soovitava intervalliga.

Edasi toimus relee juhtimine läbi Raspberry Pi GPIO väljundkanalite. Kuna relee on loodud kasutama 5V sisendpinget, siis tuli lisaks GPIO ühendustele ka eraldi 5v toide ja maandus GND Raspberry Pi-ga ühendada. Relee seadistamisel on 2 võimalust. Normaalolekus avatud vooluahel (NO) ja normaalolekus suletud vooluahel (NC). Normaalolekus avatud vooluahela puhul on sisend- ja väljundühendused tavaolekus ühendamata. See tähendab, et relee impulsi läbi luuakse ühendus ja tekib vooluring. Normaalolekus suletud ühenduse puhul on tavaolekus sisend ja väljund ühendatud. GPIO impulsi korral katkestab relee vooluringi. Käesoleva bakalaureusetöö käigus sai relee ühendatud normaalolekus avatud asendis. Vastavas Python juhtimiskoodis on võimalik määrata, milline GPIO kanal annab signaali

relee käivitamiseks. Selle abil on võimalik juhtida 8 erinevat releed läbi GPIO paneeli. Juhtimisprogrammi käivitamine on kajastatud joonisel 6.

Juhtarvutisse oli tarvis laadida tarkvara, mis kogu seadme tööd juhiks. Arm8 arhitektuuril põhineval juhtarvutil on võimalik kasutada erinevaid programmeerimiskeeli. Peamisteks on Python, C++, Scratch, HTML. Käesolevas töös osutus valituks Python, kuna suur hulk õppematerjale ja erinevaid projekte on vabavarana saadaval. Samuti on saadaval juhiseid erinevate seadmete seadistamiseks ja käivitamiseks koostöös Raspberry Pi-ga.

Töö autori eelnev programmeerimiskogemus on olnud minimaalne. Selle puudujäägi korvamiseks uuriti erinevaid internetis leiduvaid õpetusi Python programmeerimiskeele alustest. Samuti uuriti erinevaid releede ja temperatuuriandurite ühendamisskeeme ja nende juhtimiseks loodud Python koode. Paljud probleemid on leidnud lahenduse ka lihtsalt otsingumootori abil teiste probleempüstitusi uurides.



Joonis 6. Seadme käivitamine optimeerimisprogrammi tarbeks

4. JUHTIMISE TEOSTAMINE

Juhtimisprogramm koostati Python programmeerimiskeeles. Raspberry Pi juhtarvuti jaoks valiti Raspberry Pi OS operatsioonisüsteem, mis laeti MicroSD mälukaardile. Operatsioonisüsteemi installeerimisega paigaldatakse lisaks ka palju vajaminevaid programme. Näiteks Thonny Python, VNC ja muud. Optimeerimisprogramm kirjutati Python programmeerimiskeeles ja erinevate moodulite jaoks valiti paketihaldur Poetry. Optimeerimisprogramm käivitub Nord Pool kataloogis terminali lisatud käsuga: poetry run python main.py. Koodi soovitud parameetreid saab muuta seadme kaustas vastava nimega failis.

4.1. Nord Pool Spotist informatsiooni hankimine ja töötlemine

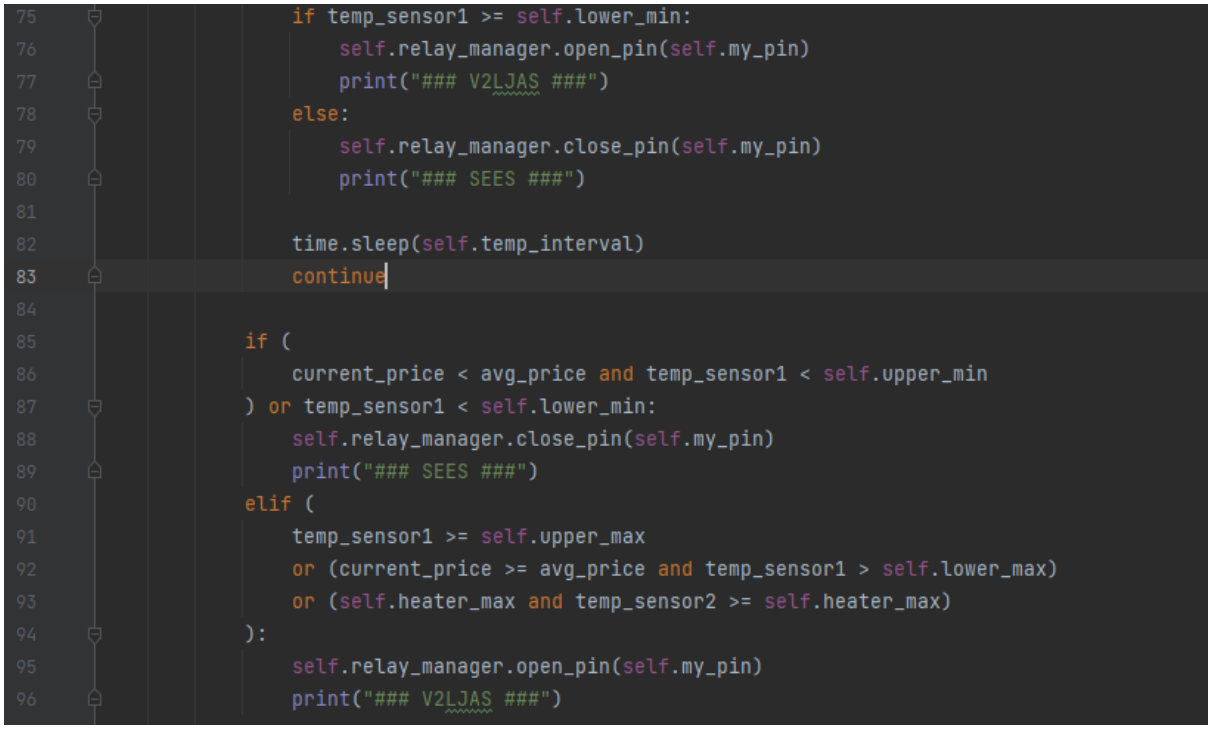
Optimeerimisprogrammi käivitamisel laeb programm alla Nord Pool Spot hinnainformatsiooni. Vastav python koodiga programm hangib vastava hinnainformatsiooni Nord Pool Spot leheküljelt. Edasi salvestatakse keskmised ja tunnipõhised hinnad nordpool_data.jsno faili. Python programm on seadistatud kasutama päeva keskmist hinda ja see näit salvestatakse koos tunnihindadega json faili.

```
94      def _parse_nordpool_json(self, data: dict):
95          timestamp = datetime.datetime.now()
96          result = dict({"timestamp": str(timestamp), "data": dict()})
97          try:
98              for index, row in enumerate(data["data"]["Rows"]):
99                  for column in row["Columns"]:
100                     day = column["Name"]
101                     if index < 24:
102                         if day not in result["data"]:
103                             result["data"][day] = dict({"hour_data": dict()})
104                         result["data"][day]["hour_data"][index] = float(
105                             column["Value"].replace(",", ".").replace(" ", "")
106                         )
107                     elif index == 26:
108                         result["data"][day]["avg"] = float(
109                             column["Value"].replace(",", ".").replace(" ", "")
110                         )
111             return result
112         except Exception as e:
113             raise Exception(f"Failed to parse nordpool data. Exception: {e}")
```

Joonis 7. Lõik Python koodist: Parse-funktsiooni kasutamine sobiva info hankimiseks

4.2. Juhtseadme otsuse tegemine

Selleks, et optimeerimine oleks võimalik, arvestab juhtarvuti iga päeva lõikes keskmise hinna. Programm loeb Nord Pool Spot andmetest antud kellaaja kohta kehtivat hinda, kõrvutab selle päeva keskmise hinnaga ja otsustab neid tegureid arvesse võttes, kas käivitada kütteseade või mitte. Programm on seadistatud nii, et kui otsustuse hetkel olev tunnihind on keskmisest odavam, siis seade kütab ning kui hind on kõrgem, siis ei küta. Lisaks on seadmele määratud hõlpsasti muudetav temperatuuriline piirang, mis sätestab punkti, mille puhul ka odavama hinna puhul seade lõpetab kütmise. Kuna seade võiks idee poolest asendada täielikult klassikalist termoregulaatorit, siis sai seadmele lisaks paigaldatud ka alumine piirväärtus. Ehk kui temperatuur langeb madalamale teatud temperatuurist, siis seade kütab ka keskmisest kallima hinna puhul. See lisa sai koodi paigaldatud juhuks, kui kallima perioodi vältel ruumi periood langeb soovitud temperatuurist madalamale. Börsihinna puhul võib olla hetkehind kõrgem keskmisest hinnast väga pika perioodi vältel. Sellest tulenevalt võib köetava ruumi temperatuur langeda soovitud temperatuurist oluliselt madalamale. Seadme kasutamine ei tohiks olla kompromiss mugavuse arvelt. Optimeerimisprogramm on seadistatud nii, et ta hoiaks soovitud temperatuurist keskmist temperatuuri 1 °C kõrgemal odavatel aegadel ja rakendaks kütte kallimatel aegadel siis, kui keskmine temperatuur langeb 1 °C alla soovitud temperatuuri. Nii on tagatud võimalikult väikeste temperatuuri erinevustega küttereźiim, mis tagab ruumi ühtlase sisekliima. Vältimaks seadme asjatut sisse-välja-lülitumist, sai seadistatud temperatuurid 2 kraadilise erinevusega. Kui seade hoiab keskmist temperatuuri 25 °C, siis seade lülitab kütte järele kui temperatuur langeb alla 24 °C. Kütmise lõpetab seade siis, kui temperatuur ületab 26 °C. Juhtseadmele on lisaks paigaldatud teine temperatuurisensor. Selle abil lülitatakse seade välja suveperioodil, kui põrandakütte töö pole vajalik. Käesoleva töö raames oli välistemperatuuri sensor seadistatud nii, et üle 20 °C välisõhu temperatuuri juures seade mitte mingil juhul ei küta. Seda piirpunkti on võimalik hõlpsasti muuta Python-koodis. Kindlasti tuleb arvestada sellega, et temperatuuriandur ei asuks otsese päikese käes.

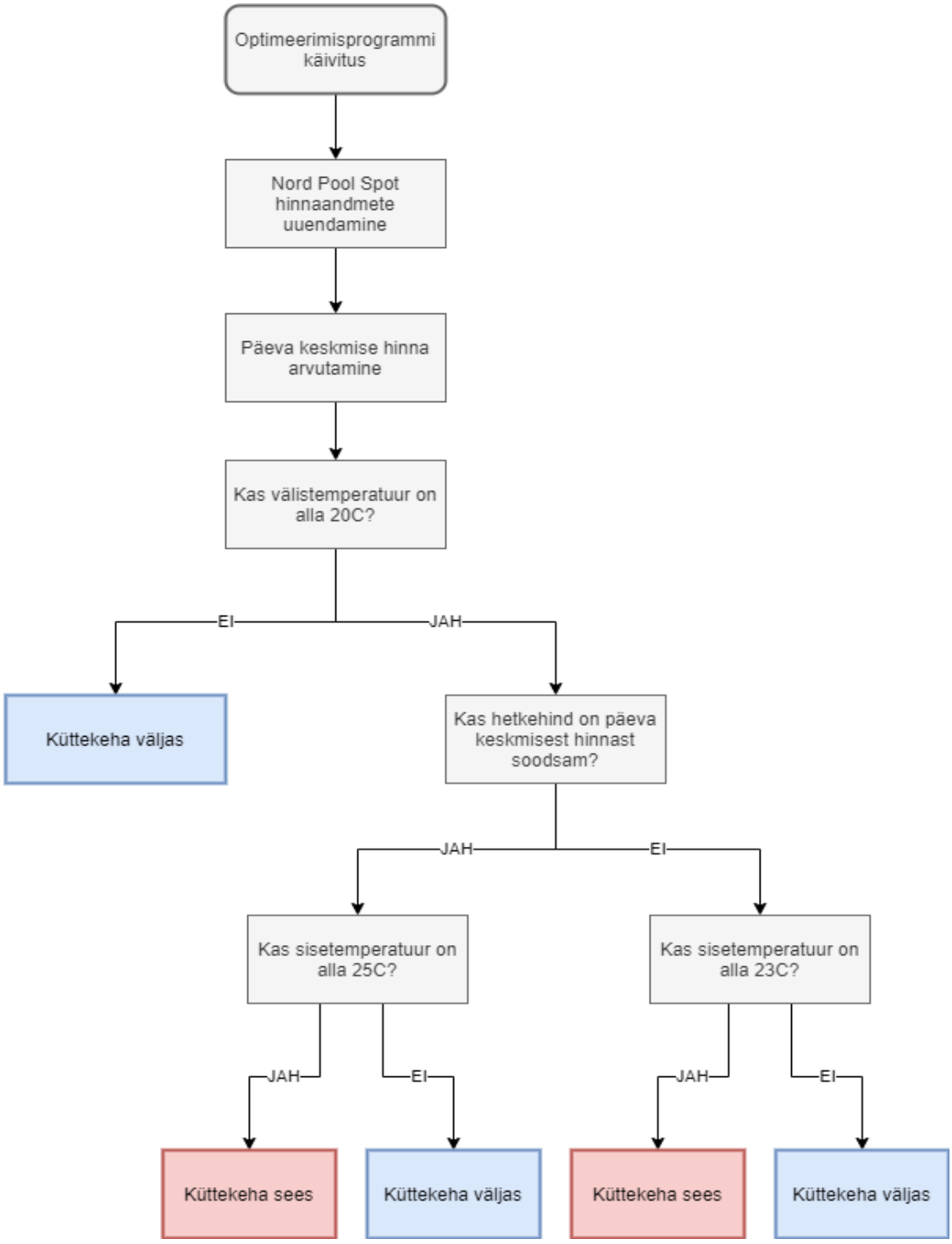


```
                if temp_sensor1 >= self.lower_min:
                    self.relay_manager.open_pin(self.my_pin)
                    print("### V2LJAS ###")
                else:
                    self.relay_manager.close_pin(self.my_pin)
                    print("### SEES ###")

                time.sleep(self.temp_interval)
                continue

            if (
                current_price < avg_price and temp_sensor1 < self.upper_min
            ) or temp_sensor1 < self.lower_min:
                self.relay_manager.close_pin(self.my_pin)
                print("### SEES ###")
            elif (
                temp_sensor1 >= self.upper_max
                or (current_price >= avg_price and temp_sensor1 > self.lower_max)
                or (self.heater_max and temp_sensor2 >= self.heater_max)
            ):
                self.relay_manager.open_pin(self.my_pin)
                print("### V2LJAS ###")
```

Joonis 8. Osa mis reguleerib relee käivitamist, arvestades temperatuuri ja keskmist hinda

4.3. Otsuse põhjal relee käivitamine/sulgemine

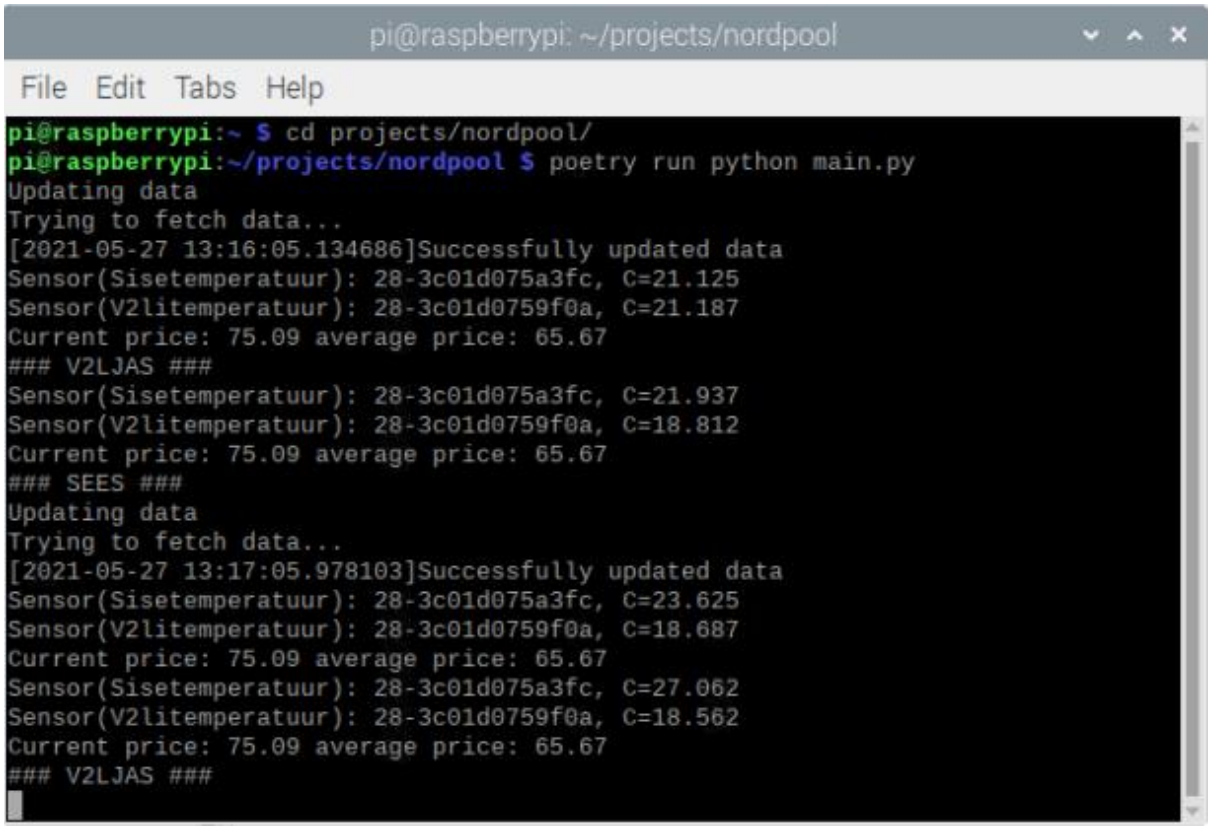
Kütteprogramm on seadistatud hankima ruumi temperatuuri informatsiooni iga 30 sekundi järel ja seejärel otsustama, kas lülitada küte sisse või mitte iga 60 sekundi järel. Kui optimeerimisprogramm otsustab kütte sisselülitamise, siis annab seade GPIO 27 kanali vahendusel impulsi, mis rakendab relee ja ühendab vooluringi. Seeläbi lülitatakse kütteseade sisse ja toimub elektrienergia tarbimine. Kui temperatuurisensor annab signaali, et ruumi programmis ettemääratud temperatuur on saavutatud, annab seade releele käskluse vooluahel katkestada. Seade kontrollib ruumi temperatuuri jooksvalt ja peale määratud alumise väärtuseni jahtumist taasalustab kütmist. Joonisel 9 on kajastatud seadme optimeerimisprogrammi juhtloogika.



Joonis 9. Seadme optimeerimisprogrammi juhtloogika

Optimeerimisprogramm on seadistatud näitama iga 60 sekundi järel temperatuuri andmeid, vastava tunni elektrihinda, antud päeva keskmist elektrihinda ja otsust, kas kütta või mitte. Kogu see info kuvatakse terminal-programmis, kui seade töötab. Selle info kuvatõmmis on nähtaval joonisel 10. Selle abil on võimalik reaalajas näha seadme tööd. Python programm

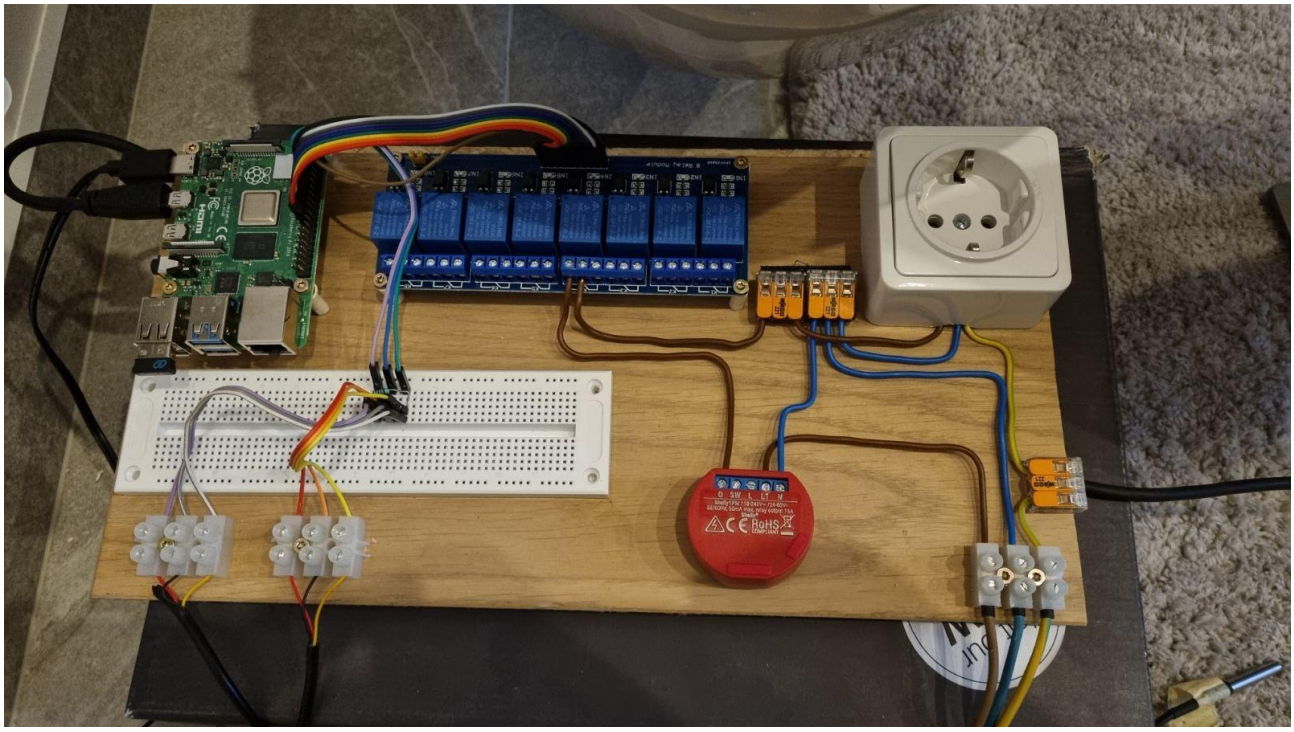
arvestab Nord Pool Spotist saadud informatsiooni. See tähendab, et kajastuvad hinnad ei sisalda Eestis arvestatavat käibemaksu ja elektrimüüja marginaali.



Joonis 10. Optimeerimisprogrammi käivitamisel kuvatud informatsioon

5. TULEMUSTE ANALÜÜS

Tulemuste saamiseks testiti seadet korterelamu vannitoas. Seade ühendati põrandaküttega ja mõõdeti energia tarbimist. Et oleks võimalik võrrelda seadmeta kulusid, sai sama tehtud ka ilma optimeerimisseadmeta. Mõlema vaatluse puhul kasutati Shelly 1PM energiakulu mõõtjat. Eelnimetatud seade kogus informatsiooni hetkelistest väärtustest ja mõõtis kogu energiakulu. Enne mõõdistusi sai põrandaküttе 25 °C juures hoitud 24 h. See tagas selle, et mõlema mõõtmise puhul ei läheks suur hulk energiat ruumi üles kütmiseks. Lõplik seade on näha alljärgneval joonisel 11.

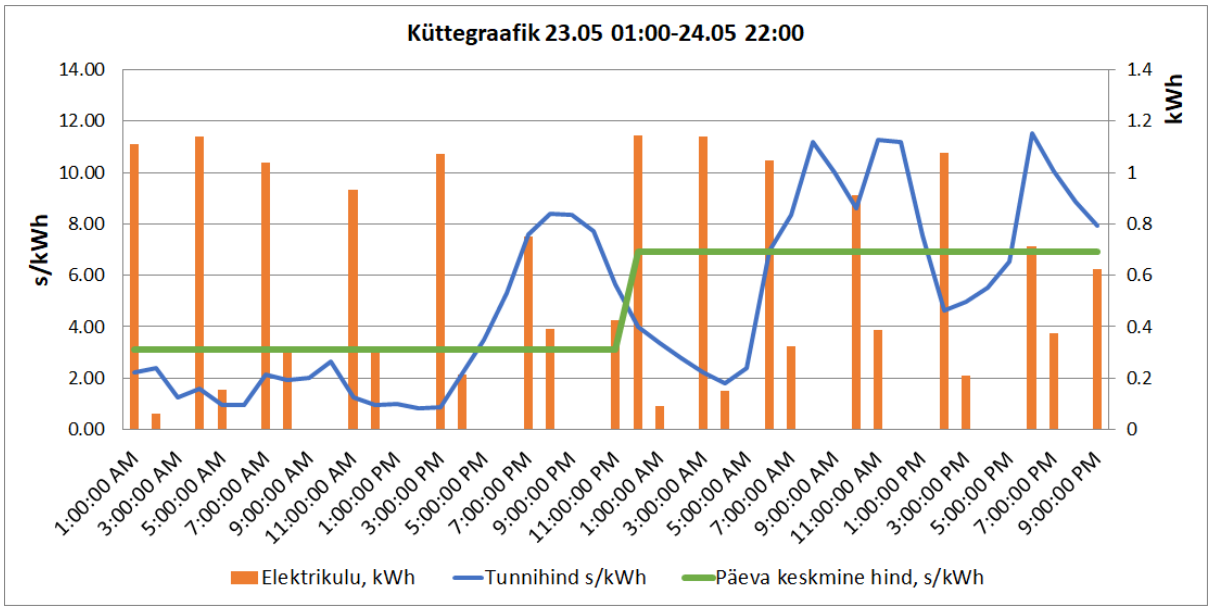


Joonis 11. Lõplik koostatud seade

5.1. Energiakulu seadmeta

Põrandaküttе tavaoleku informatsiooni hankimiseks ühendati vooluvõrku Shelly 1PM energiakulu mõõtja. Selle abil oli võimalik näha, kuidas toimib põrandaküttе termostaat. Põrandaküttе energia tarbimist mõõdeti 23.05 01:00 kuni 23.05 22:00. Elektrienergia mõõtja tuvastas, millistel perioodidel seade toimis ja kui palju energiat tarbis. Seade oli sätestatud hoidma temperatuuri 25,5 °C. Seadme talletatud informatsioonilt lähtub, et termostaat hoidis

küttekaablit järel umbes iga kahe tunni tagant tund aega, et saavutada seadistatud temperatuur. Selle temperatuuri hoidmiseks kulutati 45 tunni vältel 16,12 kWh energiat. Põrandakütte võimsuseks koos termostaadiga näitas Shelly 1PM 1.14 kw-1.16 kw. Hetkedel, mil seade põrandakütet järel ei hoidnud, luges mõõtja 43 W voolutarvet. See tuli termostaadi digitaalse ekraani töös hoidmisest sellel perioodil. Küttegraafik on näha joonisel 12.

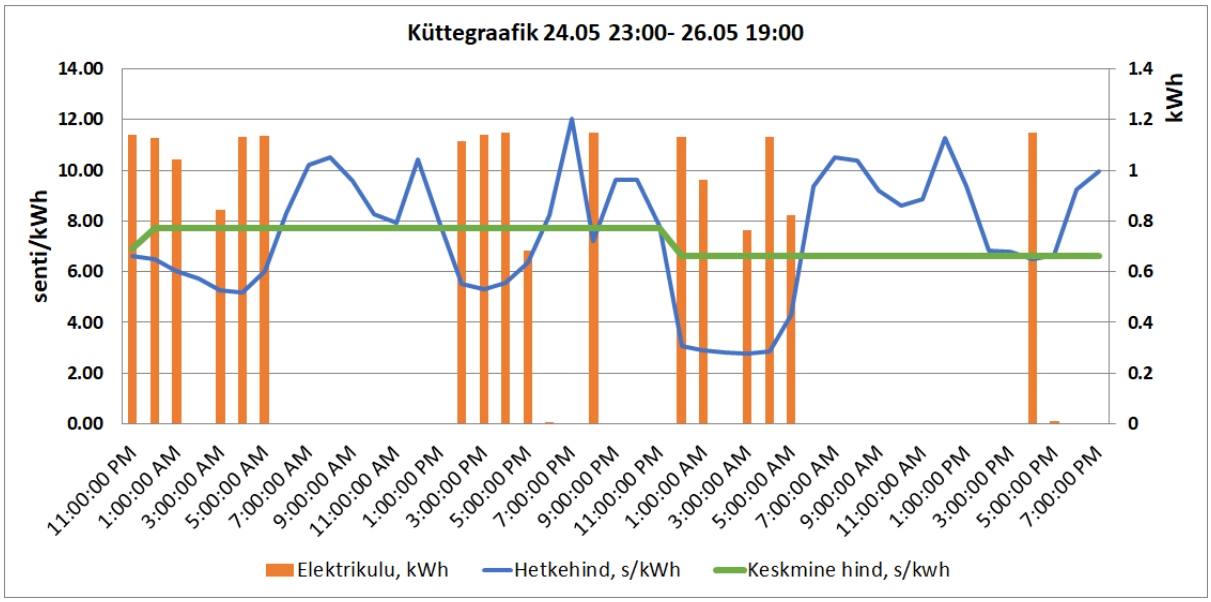


Joonis 12. Küttegraafik termostaadiga kütmisel

5.2. Energiakulu seadmega

Seade ühendati elektrikilbi ja põrandakütte termostaadi vahele alates 24.05 23:00 kuni 26.05 19:00. Küttegraafik on nähtaval joonisel 13. Seade toimis ja optimeeris kütteseadme tööd 45 tundi. Selle perioodi vältel kogus ja salvestas informatsiooni relee ja elektrikilbi vahele ühendatud Shelly 1PM elektrikulu mõõtja. Energiakulu mõõtja salvestas hetke võimsuseks küttekehal 1.14 W - 1.16 W. Käesoleva vaatlusperioodi lõikes tarvitas kütteseade kokku 17,62 kWh elektrit. Seadme tarbimise karakteristikast on näha, et optimeerimisprogramm lülitas põrandakütte võrku vaid keskmisest odavamatel börsihinnaga perioodidel. Kuna põranda betoonpinna soojuse salvestamise võime on üsna suur, siis ruum kütteta perioodil alla miinimum väärtuste ei langenud. Küttekeha maksimumpiir sai katse eesmärgil tõstetud kõrgemale kui optimeerimisprogrammis sätestatud. Nii ei seganud põrandakütte termostaat optimeerimisprogrammi tööd. Termostaat hoidis kütteseadme alati järel, kui relee läbi

vooluring tekkis. Põranda küttekeha keskmisest kallimatel perioodidel ei rakendunud. See viitab sellele, et korter on hea soojapidavusega ja üles köetud ruum soojusenergiat nii palju ei kaotanud, et oleks alla miinimumväärtuse langenud.



Joonis 13. Küttegraafik optimeerimisprogrammiga kütmisel

5.3. Majandusanalüüs

Elektrihind arvestatakse kulutatud energiahulga põhjal ja arvutatakse maksumus kWh baasil. Elektrihind jaguneb viieks osaks: võrguteenus, elektri hind, käibemaks, taastuvenergia tasu ja aktsiis.

Võrguteenus ehk elektri edastustasu määratakse võrgulepingu alusel. Sellega kaetakse kulud elektri edastamisega seotud taristu ülalpidamiseks, rikete likvideerimiseks ja tõrgeteta edastuse ülalhoidmiseks. Võrguteenus on kas ühe- või kahetariifne. See tähendab, et kas on üks fikseeritud hind või on päevane ja öine tariif eraldi. Käesolevas lõputöös kasutatakse Elektrilevi Võrk1 paketti. Selles on sätestatud võrguedastus tasu 6,14 s/kWh, mis sisaldab käibemaksu.

Elektri hind määratakse elektrienergia müügilepinguga. Eestis pakuvad elektrienergiat väga paljud. Näiteks Eesti Energia AS, Alexela AS, 220 Energia OÜ. Elektrienergia

müügilepinguga määratakse hind, millega elekter kliendile müüakse. See võib olla ühetariifiline, kahetariifiline või börsihinnal baseeruv. Käesolevas lõputöös käsitletakse just börsihinnal põhinevat elektrienerga müüki. Töös kasutatakse töö autori enda majapidamises valitud Alexela AS börsihinna paketti. Paketis sätestatud hind on kulu hetkel olev börsihind+käibemaks+marginaal. Marginaal on sätestatud börsihinna paketis: 0.36 s/kWh.

Käibemaks lisandub elektri müügi ja edastamisega seotud kuludele. Samuti lisandub käibemaks taastuvenergia tasule. Käibemaks on Eestis 20% summast. Taastuvenergia tasu on sihtotstarbeline maks, mis on loodud taastuvenergia tootjate toetamiseks. Selle abil kasutatakse saadud vahendeid taastuvenergia arendamiseks Eestis. 2021. aastal on taastuvenergia tasu 1.13 s/kWh + käibemaks.

Elektrienergia aktsiis on 01.05.2020 seisuga 1 eur/MWh. Ehk 0.1 senti/kwh. Aktsiisivabastuse loaga on elektrienergia aktsiis intensiivse tarbimisega ettevõtetele 0.05 s/kWh.

5. 3. 1 Tarbimisele kulunud summad

Kogu elektrienergia summa, mis kulus testperioodil põrandaküte kütmiseks, on välja toodud tabelis 7.

Tabel 7. Elektrienergia peale kulunud summa ilma optimeerimisprogrammita

Börsihinnale kulunud summa	75 senti
Võrguedastustasud	98 senti
Taastuvenergia tasu	22 senti
Aktsiis	2 senti

Kui arvestada kõiki kulusid, siis maksis 16,12 kWh elektrienergiat 1 euro ja 97 senti. Summad sisaldavad käibemaksu. Sama pikkusega perioodi optimeeritud kütmisel suurenes vähesel määral ka elektrienergia tarbimine. Energiakuludele vastavad rahalised väärtused väljenduvad tabelis 8.

Tabel 8. Elektri tarbimisele kulunud summa python juhtprogrammi rakendamisel:

Börsihinnale kulunud summa	92 senti
Võrguedastustasud	1 euro ja 8 senti
Taastuvenergia tasu	24 senti
Aktsiis	2 senti

Kõik kulud sisse arvestatuna kulus 17.62 kWh energia tarbimisele 2 eurot ja 26 senti. Summad sisaldavad käibemaksu.

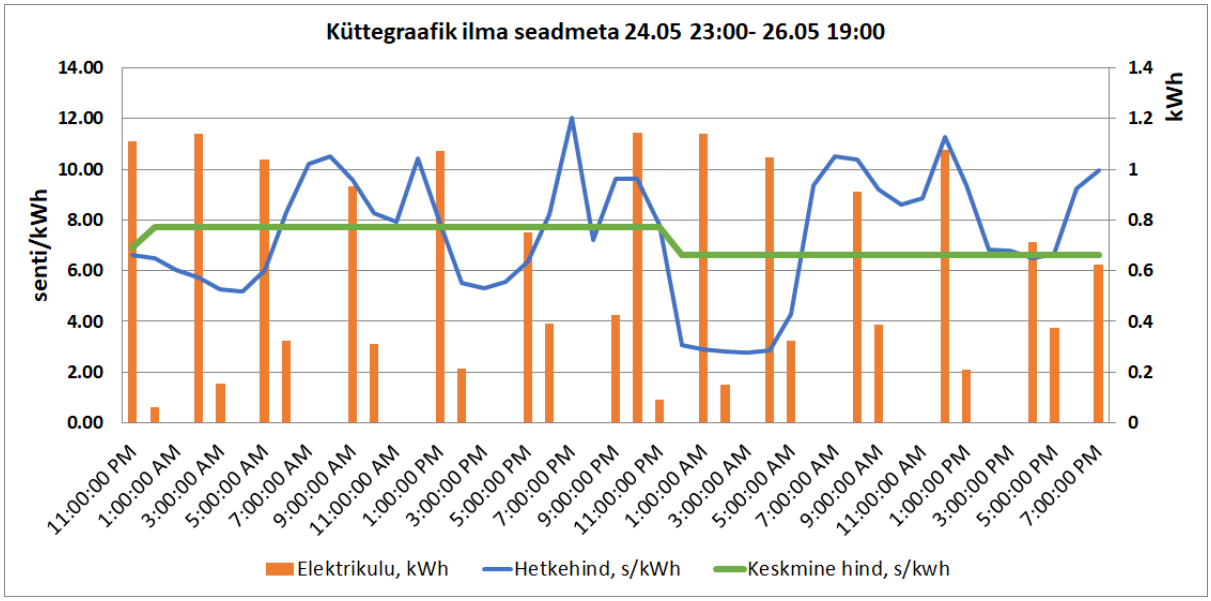
Kahe eraldi mõõdistuse ajaperiood oli erinev. Sellest tulenevalt olid ilma optimeerimisprogrammita mõõtmisel ühel päeval hinnad keskmiselt pea kolm korda odavamad kui teisel. See andis efekti, et ilma optimeerimiseta ajavahemikul oli üldjoontes odavam elekter.

Kui asendada seadmeta energiaandmed perioodile, mil mõõdeti optimeerimisseadmega, siis oleks tulemused järgnevad:

Tabel 9. Elektrile kulunud summa, kui asendada andmed juhtseadme mõõdistusperioodi

Börsihinnale kulunud summa	1 euro 18 senti
Võrguedastustasud	98 senti
Taastuvenergia tasu	22 senti
Aktsiis	2 senti

Kui arvestada kõiki kulusid, siis maksis 16,12 kWh elektrienergiat 2 eurot ja 40 senti. Summad sisaldavad käibemaksu.
Järgnev joonis kajastab asendamist iseloomustavat graafikut.



Joonis 14. Seadmeta küttegraafik ajavahemikus 24.05 23:00-26.05 19:00

Eelnevatest andmetest selgub, et samal ajahetkel oleks vastavate börsihindadega kütmine olnud ilma seadmeta oluliselt kulukam.

KOKKUVÕTE

Lõputöö käsitles kütteseadme toimivust börsihinna kõikumisi arvesse võttes. Töö eesmärgiks oli koostada elektrienergia börsihinda arvestav elektrikütte juhtseade. Koostatud seade kogus edukalt Nord Pool Spot keskkonnast börsihinna informatsiooni, töötles selle edaspidiseks kasutamiseks sobivasse formaati ja talletas selle nordpool_data.json faili. Peale Python juhtprogrammi käivitamist uuendab seade informatsiooni ning arvestab päeva keskmist ja hetkehinda. Seejärel otsustab, kas lülitada seadme järel olev põrandakütte seade sisse või mitte. Optimeerimisprogramm oli seadistatud kütteseadet järele lülitama vaid keskmisest odavamate hindade puhul. Välja arvatud juhul, kui temperatuur langeb liiga madalale. Selleks oli seadmel seadistatud minimaalväärtus, mille korral seade alustas ka kütmist kallima hinna korral. Kogu seade ja juhtprogramm oli seadistatud kasutamiseks külmematel perioodidel. Seetõttu sai lisatud seadmele ka teine termoandur, mis mõõdab välisõhu temperatuuri. Teine temperatuurianduri signaali abil lülitab seade kütmise välja, kui välisõhu temperatuur ületab 20 °C.

Juhtseade ühendati testperioodiks korterelamu vannitoa põrandaküttega. Seade juhtis põrandakütte tööd vastavalt börsihindadele. Testperioodi vältel koguti informatsiooni kulutatud elektrienergia kohta vastavalt kellaajale. See võimaldas kõrvutada energiaandmed börsihindadega ja kalkuleerida põranda kütmisele kuluv summa. Energiakulu mõõtja abil saadud info näitas, et börsihinda jälgiv seade kulutas 9,3 % rohkem energiat kui ilma optimeerimata põrandaküte kulutas samal ajahulgal. Optimeerimisega kulus 45-tunnisel testperioodil elektrienergiale kokku 2 eurot ja 26 senti.

Programmi juhtimise rakendamise abil oli võimalik näha, et tulemusi mõjutab väga suurel määral börsihindade suurus ja ajavahemik, millal hinnad kõrged on. Kui võrrelda börsihinna põhjal optimeeritud kütmist optimeerimata kütmisega, siis peab kindlasti arvestama, et eri perioodide hinnad kõiguvad väga suurel määral ja see mõjutab lõplikult energiale kujunenud hinda. Seda arvestades sai kõrvutatud optimeerimisseadmeta andmed samade börsihindadega, mis olid juhtseadme testil. Seda arvesse võttes, oleks sama perioodi juhtimata regulaatiori küttekulu 6.17% suurem juhitud põrandakütte ees.

Töö eesmärk oli koostada seade, mis arvestaks börsihinda. Seade toimis edukalt börsihinda arvestava kütteseadmena. Küttekulude enamaks vähendamiseks on võimalik juhtprogrammi veelgi optimaalsemaks seadistada.

KASUTATUD KIRJANDUS

1 10k Ohm takisti
https://www.oomipood.ee/product/m0_6w_10k_10k_0_6w_1_met_film?q=10k%200.6W%201%25%20met.film (02.05.2021)

2 8-kanaliga relee moodul
https://www.osta.ee/8-kanaliga-relee-moodul-nt-arduinole-84052280.html (02.05.2021)

3 Eesti elektrituru areng
https://elering.ee/elektrituru-kasiraamat/3-elektriturg/31-eesti-elektrituru-areng-ja-turu-avanemisega-kaasnev-kasu (18.05.2021)

4 Eesti tarbimismaksimum https://elering.ee/elektri-tarbimine-ja-tootmine (18.05.2021)

5 Maketeerimislaud
https://www.oomipood.ee/product/pps0700_maketeerimislaud_176_46mm_700_punkti?q=Maketeerimislaud (02.05.2021)

6 Maketeerimislaua juhtmed
https://www.oomipood.ee/product/vma428_maketeerimislaua_juhtmed_40tk_isa_ema_15cm_2_54mm?q=maketeerimislaua%20juhtmed (02.05.2021)

7 Nord Pool Spot andmekogum
https://www.nordpoolgroup.com/api/marketdata/page/47 (27.05.2021)

8 Raspberry Pi 4 Model b tehnilised andmed ja joonised
https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf (24.05.2021)

9 Raspberry Pi 4 Model b hind
https://arvutitark.ee/est/tootekataloog/0/Raspberry-Pi-4-Model-B-HDMI-Cable-Micro-HDMI-To-HDMI-1m-Black-T7732AX-B-1M-500841
https://arvutitark.ee/est/tootekataloog/Arvutikomponendid-Barebone/Raspberry-Pi-4B-4x-15-GHz-2-GB-RAM-WiFi-and-BT-SoC-Mini-Mainb-3051886-452569 (12.04.2021)

10 SanDisk MicroSD
https://arvutitark.ee/est/tootekataloog/Foto-ja-videokaamerad-Malukaardid-jms-MicroSD-kaardid219/SANDISK-BY-WESTERN-DIGITAL-MEMORY-MICRO-SDXC-128GB-UHS-I-W-A-SDSUAR-128G-GN6MA-SANDISK-402790 (02.05.2021)

11 Shelly 1 PM elektrikulu mõõtja
https://dimmer.ee/toode/shelly-1pm/ (19.05.2021)

12 Shelly 1PM tehnilised andmed ja joonised
https://shelly.cloud/knowledge-base/devices/shelly-1pm/#pinout (23.05.2021)
13 Stontronics Power Adapter
https://arvutitark.ee/est/tootekataloog/Arvutikomponendid-Toiteplokid-PSU36/STONTRONICS-Power-Adapter-Raspberry-Pi-4-Model-B-PSU-51-V-3A-USB-C-UK-EU-Plug-15m-must-565526 (02.05.2021)

LIHTLITSENTS

Lihtlitsents lõputöö salvestamiseks ja üldsusele kättesaadavaks tegemiseks ning juhendaja(te) kinnitus lõputöö kaitsmisele lubamise kohta

Mina, Raido Kaju,

sünniaeg 17.08.1989,

1. annan Eesti Maaülikoolile tasuta loa (lihtlitsentsi) enda koostatud lõputöö

Elektrienergia börsihinda arvestav elektrikütte juhtseade,

mille juhendaja(d) on nooremteadur Heiki Lill, MSc,

1.1. salvestamiseks säilitamise eesmärgil,
1.2. digiarhiivi DSpace lisamiseks ja
1.3. veebikeskkonnas üldsusele kättesaadavaks tegemiseks

kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile;
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Lõputöö autor Raido Kaju
(*allkirjastatud digitaalselt*)

Tartu, 28.05.2021

Juhendaja(te) kinnitus lõputöö kaitsmisele lubamise kohta

Luban lõputöö kaitsmisele.

_____Heiki Lill_____ _____28.05.2021_
(*juhendaja nimi ja allkiri*) (*kuupäev*)

_____ _____
(*juhendaja nimi ja allkiri*) (*kuupäev)*

LISAD

Lisa 1 Optimeerimisprogrammi Python kood

```
import data
import sensor
import relay
import time
import os


class App:
    def __init__(self):
        self.url = "https://www.nordpoolgroup.com/api/marketdata/page/47"
        self.datafile = "nordpoolspot_data.json"
        self.pinlist = [2, 3, 17, 27, 22, 10, 9, 11]
        self.my_pin = 27

        self.temp_interval = 60  # seconds
        self.poller_interval = 3600

        self.upper_max = 27
        self.upper_min = 25

        self.lower_max = 25
        self.lower_min = 23

        self.heater_max = 20

        self.data_manager = None
        self.relay_manager = None
        self.sensor_manager = None

    def run(self):
        self.data_manager = data.DataManager(self.url, self.datafile)
        self.relay_manager = relay.RelayManager(self.pinlist)
        self.sensor_manager = sensor.SensorManager()

        self.data_manager.start_poller(self.poller_interval)
```

```
        sensor1 = self.sensor_manager.search_sensor_name("3FC")  # room temp
        sensor2 = self.sensor_manager.search_sensor_name("F0A")  # heater
temp

        if not (sensor1 and sensor2):
            raise Exception(f"Unable to find all sensors. Found: {sensor1}
{sensor2}")

        while True:
            temp_sensor1 = None
            temp_sensor2 = None

            # get sensor1 (room) temp
            try:
                temp_sensor1 = self.sensor_manager.read_sensor_temp(sensor1)
                print(f"Sensor(sisetemperatuur):                    {sensor1},
C={temp_sensor1:,.3f}")
            except Exception as e:
                print(
                    f"Error: Failed to read temp. Waiting 60sec before
retry. Exception: {e}"
                )
                time.sleep(60)
                continue

            # if heater_max set, get heater temp
            if self.heater_max:
                try:
                    temp_sensor2                                          =
self.sensor_manager.read_sensor_temp(sensor2)
                    print(f"Sensor(v2listemperatuur):              {sensor2},
C={temp_sensor2:,.3f}")
                except Exception as e:
                    print(f"Failed to get heater temp. Exception: {e}")

            # get current and average prices
            try:
                current_price = self.data_manager.get_current_price()
                avg_price = self.data_manager.get_avg_price()
                print(f"Current price: {current_price} average price:
{avg_price}")
```

```
            except Exception as e:
                print(
                    f"Error: Failed to get prices from data. Falling back to
keep temp at lower minimum. Exception: {e}"
                )
                # fall back to heat only when room temp below lower minimum
                if temp_sensor1 >= self.lower_min:
                    self.relay_manager.open_pin(self.my_pin)
                    print("### V2LJAS ###")
                else:
                    self.relay_manager.close_pin(self.my_pin)
                    print("### SEES ###")

                time.sleep(self.temp_interval)
                continue

            if self.heater_max and temp_sensor2 >= self.heater_max:
                self.relay_manager.open_pin(self.my_pin)
                print("### V2LJAS ###")
            elif (
                current_price < avg_price and temp_sensor1 < self.upper_min
            ) or temp_sensor1 < self.lower_min:
                self.relay_manager.close_pin(self.my_pin)
                print("### SEES ###")
            elif temp_sensor1 >= self.upper_max or (
                current_price >= avg_price and temp_sensor1 > self.lower_max
            ):
                self.relay_manager.open_pin(self.my_pin)
                print("### V2LJAS ###")

            time.sleep(self.temp_interval)

    def cleanup(self):
        print("\nCleaning up\n")
        try:
            self.relay_manager.cleanup()
            self.data_manager.cleanup()
        except Exception as e:
            print(e)
```

```
def main():
    app = App()
    try:
        app.run()
    except KeyboardInterrupt:
        app.cleanup()
    finally:
        print("Exiting program...")


if __name__ == "__main__":
    main()
```

Lisa 2 Töös kasutatud elektrienergia hinnad Nord Pool Spot andmebaasist

Hinnad ei sisalda käibemaksu ja elektrienergia müüja marginaali.

Tabel 2.1 Elektrienergia hinnad Nord Pool Spot andmebaasist

	26-05-2021	**25-05-2021**	**24-05-2021**	**23-05-2021**
00 - 01	22,62	50,95	30,45	23,24
01 - 02	21,26	47,32	25,02	15,47
02 - 03	20,51	44,72	20,22	16,82
03 - 04	20,25	40,93	15,65	7,35
04 - 05	20,69	40,03	11,97	10,31
05 - 06	32,77	47,37	17,03	5,08
06 - 07	75,00	66,40	55,05	5,03
07 - 08	84,59	81,98	66,44	14,96
08 - 09	83,56	84,65	90,04	13,16
09 - 10	73,57	76,79	80,51	13,74
10 - 11	68,87	66,00	68,84	18,91
11 - 12	70,78	63,07	90,76	7,29
12 - 13	91,00	83,98	90,30	4,99
13 - 14	75,07	62,34	60,06	5,23
14 - 15	53,86	43,08	35,76	4,06
15 - 16	53,56	41,29	38,58	4,37
16 - 17	51,06	43,29	43,00	15,12
17 - 18	52,35	50,09	51,44	26,13
18 - 19	73,93	65,48	93,18	41,17
19 - 20	80,00	97,22	80,54	60,27
20 - 21	30,90	57,07	70,71	66,81
21 - 22	30,86	77,19	62,92	66,75
22 - 23	28,72	77,09	60,54	61,18
23 - 00	39,99	62,04	52,27	43,92

Lisa 3 Nord Pool Spot hinnaandmete töötlemise kood

```
import urllib.request
import json
import datetime
import threading
import time
import socket
import pytz

# timeout in seconds
req_timeout = 10
socket.setdefaulttimeout(req_timeout)

class DataManager:
    def __init__(self, url: str, filename: str):
        self.url = url
        self.filename = filename
        self.last_poll = None
        self.poller = None

        self.update_data()

    def update_data(self):
        print("Updating data")
        raw_data = None
        try:
            raw_data = self._read_data_from_url()
        except Exception as e:
            print(f"Error: {e}")
            return

        parsed_data = None
        try:
            parsed_data = self._parse_nordpool_json(raw_data)
        except Exception as e:
            print(f"Error: {e}")
            return

        try:
            self._write_data(parsed_data)
        except Exception as e:
            print(f"Error: {e}")
            return

        self.data = parsed_data
        self.last_poll = parsed_data["timestamp"]
        print(f"[{self.last_poll}]Successfully updated data")
        return True

    def _read_data_from_url(self):
        req = urllib.request.Request(self.url)
        max_retries = 5
        retries = 0
        response = None
```

```
        while not response and retries < max_retries:
            try:
                print("Trying to fetch data...")
                response = urllib.request.urlopen(req)
            except Exception as e:
                print(f"Failed to fetch data. Retries: {retries}, max:
{max_retries}")
                time.sleep(10)
                retries += 1

        if response:
            try:
                return json.loads(response.read())
            except Exception as e:
                raise Exception(f"Failed to load response into json.
Exception: {e}")
        else:
            raise Exception("Failed to fetch data. Max retries limit
crossed.")

    def _write_data(self, data: dict):
        out_file = None
        try:
            out_file = open(self.filename, "w")
            json.dump(data, out_file, indent=2)
        except Exception as e:
            raise Exception(f"Failed to write data to file. Exception: {e}")
        finally:
            out_file.close()
        return out_file

    def _read_file(self):
        data = None
        try:
            in_file = open(self.filename, "r")
            data = json.load(in_file)
        except Exception as e:
            raise Exception(f"Failed to read from file. Exception: {e}")
        finally:
            in_file.close()
        return data

    def _parse_nordpool_json(self, data: dict):
        timestamp = datetime.datetime.now()
        result = dict({"timestamp": str(timestamp), "data": dict()})
        try:
            for index, row in enumerate(data["data"]["Rows"]):
                for column in row["Columns"]:
                    day = column["Name"]
                    if index < 24:
                        if day not in result["data"]:
                            result["data"][day]     =     dict({"hour_data":
dict()})
                        result["data"][day]["hour_data"][index] = float(
                            column["Value"].replace(",",  ".").replace("  ",
"")
                        )
                    elif index == 26:
                        result["data"][day]["avg"] = float(
```

```
                            column["Value"].replace(",",  ".").replace("  ",
"")
                        )
            return result
        except Exception as e:
            raise  Exception(f"Failed  to  parse  nordpool  data.  Exception:
{e}")

    def _parse_timestamp(self, timestamp):
        day = timestamp.strftime("%d-%m-%Y")
        hour = timestamp.hour
        return day, hour

    def get_current_price_by_timestamp(self, timestamp: datetime):
        day, hour = self._parse_timestamp(timestamp)
        result = None
        try:
            result = self.data["data"][day]["hour_data"][hour]
        except Exception as e:
            raise Exception(f"Unable to get price value. Exception: {e}")
        return result

    def get_avg_by_timestamp(self, timestamp: datetime):
        day, hour = self._parse_timestamp(timestamp)
        result = None
        try:
            result = self.data["data"][day]["avg"]
        except Exception as e:
            raise Exception(f"Unable to get price avg. Exception: {e}")
        return result

    def get_current_datetime_for_cest(self):
        tz_cest = pytz.timezone("Europe/Paris")
        tz_eest = pytz.timezone("Europe/Tallinn")
        now = datetime.datetime.now()
        now_local = tz_eest.localize(now)
        now_cest = now_local.astimezone(tz_cest)
        return now_cest
    def get_current_price(self):
        timestamp = self.get_current_datetime_for_cest()
        return self.get_current_price_by_timestamp(timestamp)

    def get_avg_price(self):
        timestamp = self.get_current_datetime_for_cest()
        return self.get_avg_by_timestamp(timestamp)

    def start_poller(self, interval: float = 3600):
        start = time.time()
        self.poller = threading.Timer(
            interval - ((time.time() - start) % interval), self.update_data)
        self.poller.start()

    def stop_poller(self):
        if self.poller:
            self.poller.cancel()
    def cleanup(self):
        try:
            self.stop_poller()
        except:
            pass
```